Mobility Prediction in Vehicular Networks: An Approach through Hybrid Neural Network under Uncertainty

Soumya Banerjee, Samia Bouzefrane, Paul Mühlethaler

Mobility Prediction in Vehicular Networks : An Approach through *Hybrid Neural Networks* under Uncertainty

*Soumya Banerjee †Samia Bouzefrane ‡Paul Muhethaler

Abstract

Conventionally, the exposure regarding knowledge of the inter vehicle link duration is a significant parameter in *Vehicular Networks* to estimate the delay during the failure of a specific link during the transmission. However, the mobility and dynamics of the nodes is considerably higher in a smart city than on highways and thus could emerge a complex random pattern for the investigation of the link duration, referring all sorts of uncertain conditions. There are existing link duration estimation models, which perform linear operations under linear relationships without imprecise conditions. Anticipating, the requirement to tackle the uncertain conditions in *Vehicular Network*s, this paper presents a hybrid neural network-driven mobility prediction model. The proposed hybrid neural network comprises a *Fuzzy Constrained Boltzmann machine (FCBM*), which allows the random patterns of several vehicles in a single time stamp to be learned. The several dynamic parameters, which may make the contexts of *Vehicular Networks* uncertain, could be vehicle speed at the moment of prediction, the number of leading vehicles, the average speed of the leading vehicle, the distance to the subsequent intersection of traffic roadways and the number of lanes in a road segment. In this paper, a novel method of hybrid intelligence is initiated to tackle such uncertainty. Here, *the Fuzzy Constrained Boltzmann Machine (FCBM)* is a stochastic graph model that can learn joint probability distribution over its visible units (say n) and hidden feature units (say m). It is evident that there must be a prime driving parameter of the holistic network, which will monitor the interconnection of weights and biases of *the Vehicular Network* for all these features. The highlight of this paper is that the prime driving parameter to control the learning process should be a fuzzy number, as fuzzy logic is used to represent the vague and and uncertain parameters. Therefore, if uncertainty exists due to the random patterns

*Soumya Banerjee,Visiting Prof. Conservatoire National des Arts et Metiers, Paris Cedex 03, France, Birla Institute of Technology, Mesra, India, e-mail:soumyabanerjee@bitmesra.ac.in

†Samia Bouzefrane, Conservatoire National des Arts et Metiers, Paris Cedex 03, France,

‡Paul Muhlethaler, Eva team, INRIA 2 Rue Simone IFF, 75012 Paris, France,

caused by vehicle mobility, the proposed Fuzzy Constrained Boltzmann Machine could remove the noise from the data representation. Thus, the proposed model will be able to predict robustly the mobility in VANET, referring any instance of link failure under *Vehicular Network* paradigm.

Keywords: Vehicular Network, Mobility Prediction, Link Failure, Fuzzy Constrained Boltzmann Machine, VANET, Uncertainty.

1 Introduction

With the increase of wireless networks and the growing trends towards *the Internet of Things (IoT),* vehicular communication is being viewed from different perspectives. These include the road safety and traffic management [1]. However, scenarios of vehicular networks are becoming more complex as several dynamic parameters of vehicles are being introduced : vehicle speed at the moment of prediction, number of leading vehicles, average speed of the leading vehicles, the distance to the subsequent intersection and the numbers of lanes in a road segment. The problem is thus more realistic and several research initiatives are already being accomplished, by considering the data obtained relating short-term vehicle movement [2][3]. The reliability of contexts, variables in different road intersections, different traffic scenarios and inter-vehicle link duration offer challenges to formulate the prediction model [4]. In addition to, a substantial number of research initiatives concern probabilistic modeling of vehicles which infer immediate future locations. Even so, it has been observed that to configure a robust and intelligent vehicular networks [5], each tiny parameter such as road intersection problem parameters can be handled with an effective group scheduling of vehicles. Thus those intelligent neuro-fuzzy (neural-network and fuzzy logic driven) models becoming more adaptive to suit different traffic conditions [6]. Inspired by such models [7] [8], this paper proposes a *Fuzzy Constrained Boltzmann Machine (FCBM).* This is a stochastic graph model and can learn joint probability distributions over certain time units with many existing as well as hidden features of different vehicular network environments. The relevance of proposed approach is two fold: firstly, the class of Boltzmann machine is a specialized class of deep learning algorithm and no such model currently exists. Moreover, conventional deep learning models are being controlled with visible and hidden features of problem domain. In this case, an uncertain relationship is represented with these inherent uncertainty as a fuzzy number. Thus, the constraints of relationships between the features should be driven by fuzzy logic and this could serve to train the Boltzmann machine to infer smarter decision about mobility predictions in vehicular networks. We develop the simulation and experimental model and test it with the corresponding data set. Several interesting observations have been obtained. The analysis shows that a hybrid intelligent model is required, where uncertainty and non-linear optimal conditions persist. The remaining part of the paper has been organized as follows: Section 2 briefly mentions the most relevant intelligent models deployed for vehicular networks under different conditions. Section 3 develops a mathe-

matical formulation of the proposed approach, and outlines the role of auxiliary functions for modeling fuzzy logic in Boltzmann scheme in Section 3.1. Section 4 provides a short introduction to the highlights conventional Boltzmann machine and its relevance to hybrid neural networks. Section 4.1 gives details of simulation and the corresponding results comparing them to the available data set. Finally, Section 5 highlights about the contributions and mentions future research directions in the field.

2 Related Work

Very few core research implementations are available using different computational intelligence schemes in vehicular networks for mobility prediction. Most approaches (e.g. fuzzy logic and rough set) use clustering or classifications of vehicles according to their location even in boundary regions [12] [13]. However specific intersection control problems in smart cities are being treated with neuro-fuzzy learning from real traffic conditions [14]. Traffic and vehicle speed prediction have also been developed using neural networks and hidden Markov models [6]. Inspite of all the existing models, sensing techniques and prediction of mobility in vehicular networks have raised substantial research challenges. This is primarily because, none of the intelligent models could encompass diversified uncertain parameters of vehicular networks and making the predictions unrealistic. Inspired by recent studies of deep learning and machine learning approaches [15], this paper adopts a basic Boltzmann machine approach. The model is trained through fuzzy numbers, which represents different non-linear features of vehicular network as well as network connectivity overheads. The proposed model is termed as hybrid neural network.

3 Mathematical Formulation of Proposed Model

The following parameters are being considered, while formulating the proposed model :

- Vehicle speed S_m at the moment of prediction
- Number of leading vehicles
- Average speed of leading vehicles
- Distance (optional) to the next intersection
- Number of lanes in the road segment (R_s)

These parameters are non-deterministic and lead to major concerns of uncertainty in vehicular networks. In addition, these parameters and their associated contexts can contain uncertainties. They are listed as :

- Change of vehicle Speed

Table 1: **List of Prime Variables**

Parameters/ Variables	**Semantics**
$i \in I$	Time interval between vehicles ->I= {1,2,.....m}
$j \in J$	Index of different access points (AP), where J={1,2,.....n}
γ^1	Vehicle departure ratio from source
M_{ij}	Rate of mobility from i to j
L_i	Length of time interval
x_{ij}	Mobility prediction decision variable for the points i to j
P_s	Prediction Scenario with respect to the parameters mentioned for Vehicular Network
α	time interval

- Different driving habits and road conditions
- Density of traffic
- Position of traffic lights

Therefore a specific objective function can be formulated.

The objective function is described, the objective function can train the proposed Boltzmann Machine through symmetric triangular Fuzzy Numbers. The inclusion of fuzzy factor sis to tackle uncertain parameters and their contexts mentioned in the previous description. We divide the approach into two major parts:

a. Initially, optimal control of the delay for vehicle:

The total vehicle delay time for whole network is:

$$MinV_D = [min \sum_{j \in J} \sum_{k=1}^{P_S} \sum_{i \in R_s} V_i^j(k) - M_{ij}(k)]\alpha] \tag{1}$$

where $V_i^j(k)$ is the number of vehicles for point i for road section R_s at the time instance k and $\alpha$is the sampling interval period for complete network coverage.

b. For this part, we assume that there must exist a non-linear optimal control of mobility, where, for training with the uncertain parameters of the vehicular network, a fuzzy number is introduced in triangular form (it signifies that the

core function can represent at least three values of membership or certainty factor: for example: road traffic could be moderately normal, medium, strongly adequate etc.). We also observe that there could be different trends of mobility for two communicating vehicles before the communication may fail due to the predicted enhancement in the intermediate distance. Therefore, this non-linear factor can be represented with another form of exponentiation function. Thus, if the minimum value of vehicle delay under non-linear/uncertain factors is being considered, then

$$MinV_D = [min \sum_{j \in J} \sum_{k=1}^{P_S} \sum_{i \in R_s} [V_i^j(k) - exp(\beta_0^j(k)]\alpha \tag{2}$$

Here, P_s the prediction scenario, parameter $\beta_0^j(k)$ and $V_i^j(k)$ is the result of an auxiliary function, this will be essentially to formulate a final value of the training function for the Boltzmann Machine. In practice, Boltzmann machines are comprised of visible units, (say V), and hidden units. The visible units are those which receive information from the 'environment' (in this case it could be the traffic conditions from the road and other features derived from the traffic contexts), i.e. the training set is a set of binary vectors over the set V. The distribution over the training set is denoted as a continuous function $P^+(V)$. Moreover the machine has an energy reference, which is modified through the positioning of interconnected weights of features during the operation. The machine may stop learning correctly, when it is scaled up to anything larger than a trivial value. There are two phases involved in Boltzmann machine training, and we switch iteratively between both of them. One is the positive phase, where the visible units' states are clamped to a particular binary state vector sampled from the training set (according to P^+). The other is the negative phase, where the network is allowed to run freely. Therefore, the reference energy level of the machine should be discrete out of uncertainty factors and this switching effect from positive to negative must encompass all the membership values of uncertainty [9][10]. As in this case, the hidden features in random or urban traffic conditions are free from external interference, and their values are unknown to us. Hence, we cannot update their weights. Thus, the more membership or certainty values of hidden feature vectors are introduced, the more precise prediction can be.

In the next subsection, the structure of the auxiliary function is derived.

3.1 Structure of the Auxiliary Function in a Vehicular Network

Here, we refer to the structure of $\beta_0^j(k)$:

$$\delta_0^J(k) = \Big\{ k[\gamma_0^j(k) \oplus \gamma_0^j(k+1)] \tag{3}$$

if $(1 \leqslant k \leqslant P_s - 1)$ and subsequently as the value of $\delta_0^J(k)$ could be treated as a fuzzy number trivially with 3 values ; if k= 0, then $\delta_0^J(k) = 0$, if k=P_s

then $\delta_0^J(k)$ =P_s. The construction of Right hand side of the expression with γ_0^j indicates the weight bias of fuzzy nmber which is additive with the instances of values k. That means, k and (k+1) instances are considered here for formulating auxillary function and thus β_0^j .

For implementation, we also investigate the learning features of vehicular networks and it could be either simple sample function or a multiple sample function for error estimation in the final value of the training function in the Boltzmann Machine. It is known that better scaling and an error free representation will make the network learn better for the prediction of the vehicles movement. Considering all the listed parameters, multiple sample features will be suitable to make the training of the network more error free. Assuming, the multiple sample features, the final training function, say X(w), where w is the edge weight of the features connected in the Boltzmann Machine, can be expressed as [10] :

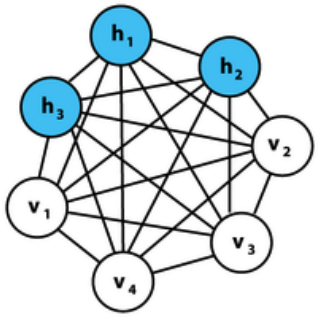
$$T(w) = \frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{m}(f_c(x^i) - y_c^i)^2 \tag{4}$$

It is clear that two terms i.e. $f_c(x^i)$ and y_c^i in the expressions related to T(w) are the coefficients of the training function to be operated on feature vectors taken from vehicular network paradigm. The first one depends on the network edge weight w where the second one is independent of w. Therefore, a partial derivative is derived for the final training weight $w_{Ps_j}^k$for all k and j.

4 Proposed High Level Description of the Hybrid Neural Network

The term hybrid neural network was coined from the concept that a neural network in its core form can be modified with the supporting mode of computational intelligence like fuzzy logic, specially to for the training purposes. Prior to describing the proposed model, a brief background is presented on conventional Boltzmann Machines (Fig. 1). They were one of the first examples of a neural network capable of learning internal representations, and are able to represent and (given sufficient time) solve difficult combinatoric problems [9]. The structure comprises of some visible and some hidden units. A graphical representation of an example of a Boltzmann machine is shown in fig. 1. Each undirected edge represents dependency. In this example there are 3 hidden units and 4 visible units.

Figure 1: **Basic Structure of Boltzmann Machine Network**



Structurally, the network can learn by the adjustment of weights and hence it finally culminates with an energy function E [9] [10]:

$$E = -(\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i)$$

where, w_{ij} is the connection strength between unit j and unit i . s_i is the state, $s_i \in \{0, 1\}$ of unit i,

θ_i is the bias of unit i in the global energy function. ($-\theta_i$ is the activation threshold for the unit).

Normally, Boltzmann machines are made of two layers of neurons, with connections only between different layers and no connections within the same layer. The bottom layer (i.e. the visible one) is denoted further by a binary vector $v = [v_1, v_2, .., v_{n_v}]$, in which each unit v_i is a binary unit, and where n_v is the size of v (i.e. the number of neurons of the visible layer). The top layer

Algorithm 1 ***Hybrid Boltzmann machine***

Given: a training set of feature vectors (n) of a vehicular network at a random condition

Assume visible neurons 10, hidden feature neuron : 625 and a standard random number function;

t= 0:

While termination condition can't be satisfied
for all features to n **do**
end for

Assign the function value as eq. (1) & (2)
Formulate T(w) following eq. (4)
for all fuzzy numbers weight w **do**
initiate final T(w)
end for
t = t+1;
end while

(i.e. the hidden one) is represented by the binary vector h = [h_1 , h_2 , .., h_{n_h}], in which each element h_j is binary, and where nh is the size of h. Furthermore, each neuron from the visible layer has associated bias. The biases of the visible neurons are grouped in a vector a = [a1 , a2 , .., an_v]. Similarly, hidden layer neurons have biases, collected in vector b = [b1, b2, .., bn_h]. The broad high level description is presented in :

In this paper, the implementations of the algorithm is done in Visual C++. In all the settings, the momentum was set to 0.5, the learning rate to 0.05. We assume 10 visible and around 625 hidden (under uncertain conditions) neurons, to be trained with a fuzzy triangular function.

4.1 Results and Analysis

The above *Fuzzy Constrained Boltzmann Machine (FCBM) algorithm* is simulated in VC++ 5.0 with MFC (Microsoft Foundation class) support and the algorithm is tested an available data set [11]. The VC++ code uses two threads that read and write from the synthesized vehicular network with the function getVecMessage ( ) and sendVecMessage( ) from the library predefined as <msn.h>. Prior to developing the desired simulation, the following propositions are made to support the simulation across the interconnected device network

- Nodes: A node is an instance of an executable and can be a sensor, actuator, processing or monitoring algorithm.
- Messages: A message is a typed data structure made up of primitive types like integer, floating point, boolean, etc., arrays of primitives and constants. Nodes communicate with each other by passing messages.
- Context of Topic: A Context of Topic is an asynchronous data transport system based on a subscribe/publish system and is identified by a name. One or more nodes are able to publish data (messages) to a topic and one

Table 2: **Analysis of Prediction Scenario with different time stamps and feature weights**

Wt. w	**0**	**2**	**3**	**4**	**5**	**6**	**7**
$\mathbf{P}_s$ **= 20**	.5 1 .86	.4 0 1	.3 .6 1	0 0 1	0 0 1	0 0 1	0 0 1
	1 .6 1	1 1 0.8	**1 1 1**	1 1 .5	1 1 .5	0 1 .7	0 1 1
	0 1 .7	1 0 1	1 .8 1	.8 0 1	1 0 1	0 0 1	1 0 1
$\mathbf{P}_s$**= 130**	.5 0 1	.6 0 0	**0 0 0**	0 0 1	0 0 1	0 0 1	0 0 1
	0 1 0	0 1 1	0 1 1	1 .4 .6	1 .6 .4	0 0 1	0 0 1
	1 .9 1	1 .9 1	**1 1 1**	.9 0 1	1 0 1	1 0 1	**1 1 1**

or more nodes can read data on that context of topic. Data is exchanged asynchronously by means of a topic and via a service. This process will help to identify more vehicular network features. Finally, this will produce a training set of vectors

- Services: A services allows nodes to communicate with each other through a synchronously communication. The service nodes are able to send a request and receive a response.

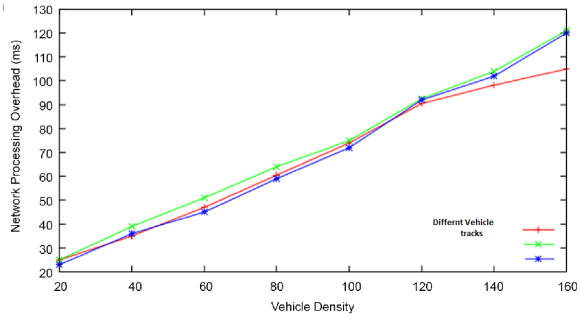
The following observation in values of weights (Fuzzy numbers) are done following the prediction scenario P_s:

The different intermediate states of the vehicular network have been demonstrated with the different weight values and there are substantial changes from weight mark iterations from 0-7. The prediction scenario P_s is also different with time stamps from 20-130 ms as shown in Fig 2. It is shown that a Fuzzy driven Boltzmann machine with different vehicle tracks and having same network overheads can predict the movement of vehicles.

The network processing overhead increases proportionally with an increase in vehicle density for all types of highway road/tracks as shown in Fig. 2. The network processing overhead in the scheme is higher because of the additional features incurred for the tasks such as accident zone identification, travel direction identification, risk factor assignment and prioritization of emergency vehicles like fire services or ambulances.

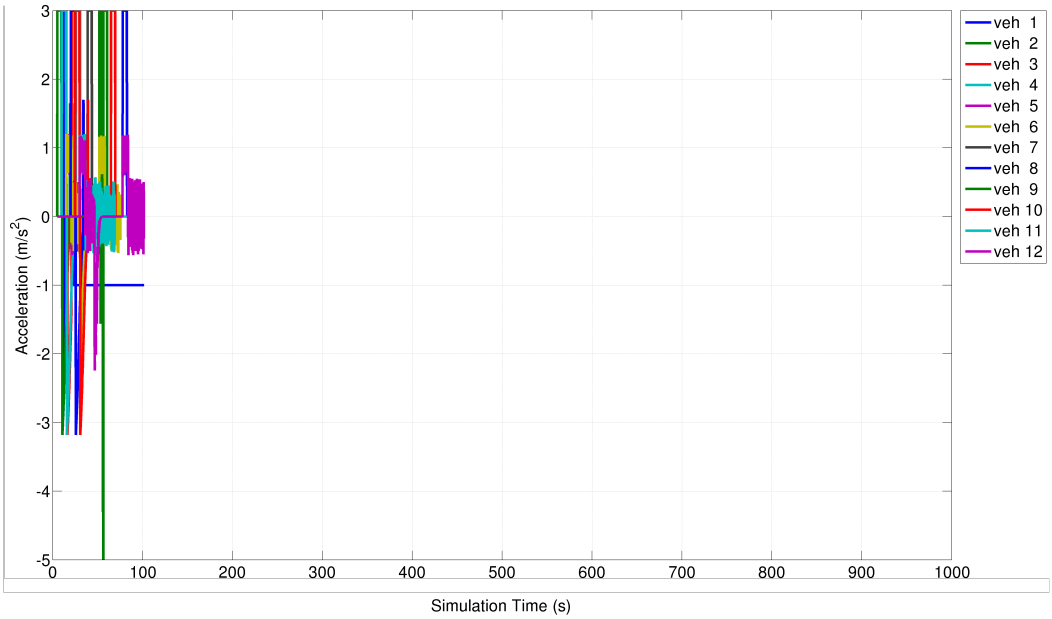
Prediction is more accurate after the first two tracks (red and blue) are being for training following the weight values shown in Table 2. The green and blue curve of the track shows certain steady value with all the features, however for prediction scenario P_s the red track diminished after certain iterations.

Figure 2: **Movement Track**



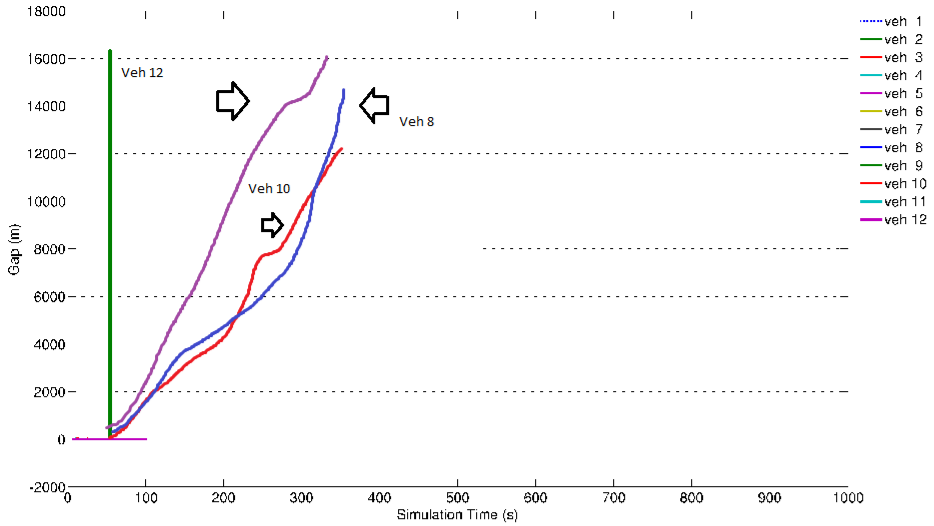
Following the data set in [11], the second part of the simulation is shown. In this case, to identify the different types of vehicles, the simulation time differs depending on the weighted feature w. However, the average vehicle density alters considerably, with respect to average connectivity distance. The other part of simulation is also done with the number and types of vehicles. The impact is one of the important parameter to understand the prediction error analysis.

Figure 3: **Impact of Vehicle acceleration and simulation time for prediction**



The plots shown in fig. 3 are closely analogous to the parameters studied: acceleration of 12 similar vehicles are being considered and the simulation time is calculated. The simulation code developed for the Boltzmann machine with fuzzy numbers as constraints, is shown in appendix, and following the code, the RMS value of the simulation performed on the data set [11] is given in terms of vehicle mobility predictions. The values collected for the phases of delivery and acknowledgement and the total time of the iteration have also been presented. For the acknowledgment phase, there are the minimum and maximum time used for the message and the number of conditional variables of the messages for vehicles through FCBM are also given. We also observe that the phases of configuring the Boltzmann machine with 10 visible and 625 hidden units can be kept as maximum to train the network. When more than 625 uncertain features from traffic conditions exist, the prediction time differs randomly, even after successful training with fuzzy triangular values.

Figure 4: **Restricted Vehicles with uncertain Gap**



In the figure 4, a plot is shown for 12 vehicles of different types, with an intermediate gap and distance between them. The plot has been restricted with iterations of simulation and it is found that, if the intermediate gap of all 12 types of vehicles are considered, then with available training scheme only vehicle number 12, 8 and 10 can be referred for effective prediction. The other vehicles cannot be considered with this training function, as the intermediate gaps are uncertain and random. The curve shows in the plot are also not smooth and it becomes more stiffer for the best convergence of vehicle 12.

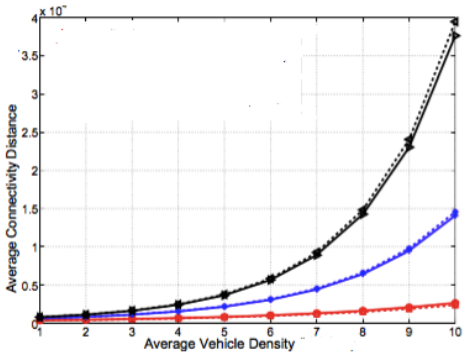
We demonstrate final results as statistical comparisons. It is the impact of vehicle density and average connectivity distance. We assume statistical Rayleigh fading with superimposed log normal scale. The results show that both vehicle density and average connectivity increases as the average vehicle density increases. Further, as shadow fading occurs, therefore, standard deviation value increases for both these parameters. This means that the average vehicle density required to satisfy a given value of average connectivity distance decreases, whenever the value of standard deviation increases. Three vehicles 12, 8 and 10 are considered to test the convergence of decision in uncertainty, and the lower red curve demonstrates minimum deviation, but with minimum fuzzy training value.

For immediate reference, we present a snapshot of the results as in Table 3: all pairs of iterations with acknowledgment and delivery of movement prediction are shown and minimum error should correspond to greater precision. It should be mentioned that, we performed a single partial derivative to obtain the final training function $T(w)$ with the auxiliary function shown in section 3.1. A higher order and more iterations of partial derivative will lead to better precision and could reduce the error value in prediction.

Table 3: **Results for Vehicle Movement**

Iteration	**Min**	**Max**	**Val. passed in function**	**RMS**	**Err.**
Ackn.	2.1	2.6	05	32.7	4.7
Delv.	1.0	7.9	15	33.6	8.7
Ackn.	5.3	8.0	20	41.2	3.92
Delv.	2.4	2.6	25	34.3	2.1

Figure 5: **Optimal Vehicle Density & Connectivity**



5 Conclusion

The paper demonstrated a novel model to predict the movement of vehicles under uncertainty conditions. The approach is implemented through a conventional Boltzmann Machine and trained with fuzzy logic and encompassing the features of a vehicular network. The hidden features and their combinations are expressed as a fuzzy triangular function and thus a computationally lightweight application could be developed. However, while deploying the simulation, it was observed that as conventionally a Boltzmann machine is used for deep learning applications (principally pattern recognition), existing Python libraries are inadequate to support the simulation. The application can be well extended with more real life data instances and if the order of partial derivation could be higher when choosing final training function, better throughput and accuracy could be obtained. A greater numbers of intelligent optimization algorithms like different variants of swarms can be chosen to select the precise combinations of

parameters. In addition, the complexity of the program may lead to a trade-off between accuracy of prediction and execution time.

References

[1] H. Omar, W. Zhuang, A. Abdrabou, and L. Li, Performance evaluation of vemac supporting safety applications in vehicular networks, IEEE Trans. Emerg. Topics Comput., vol. 1, no. 1, pp. 69–83, Jun. 2013.

[2] S.Pack and Y.Choi, Fast Handoff Scheme Based on Mobility Prediction in Public Wireless LAN Systems, in IEEE Proceedings- Communications, ,Vol. 151, Issue 5, pp. 489-495, 2004.

[3] G.Yavas, D.Katsaros, O.Ulusoy, and Y. Manolopoulos, A Data Mining Approach for Location Prediction in Mobile Environments, in Data & Knowledge Engineering, August 2005, Vol.54, Issue 2, pp. 121-146.

[4] X. Wang, C. Wang, G. Cui, and Q. Yang, Practical link duration prediction model in vehicular ad hoc networks, Int. J. of Dist. Sensor Networks, vol. 2015.

[5] Weigang Wu, Jiebin Zhang, Aoxue Luo, Jiannong Cao, Distributed Mutual Exclusion Algorithms for Intersection Traffic Control, IEEE Transactions on Parallel and Distributed Systems, Vol: 26, Issue: 1, Jan. 2015, pp.65 - 74, 2015.

[6] Jialang Cheng, Weigang Wu, Jiannong Cao, Keqin Li, Fuzzy Group-Based Intersection Control via Vehicular Networks for Smart Transportations, IEEE Transactions on Industrial Informatics,Vol: 13, Issue: 2, April 2017, pp.751-758, 2017.

[7] Nizar Alsharif, Khalid Aldubaikhy and Xuemin Shen, Link Duration Estimation using Neural Networks based Mobility Prediction in Vehicular Networks, IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE procedings, 2016.

[8] Bingnan Jiang, Yunsi Fei, Traffic and vehicle speed prediction with neural network and Hidden Markov model in vehicular networks, iEEE Intelligent Vehicles Symposium (IV), 2015 .

[9] Thomas Streubel, Karl Heinz Hoffman, Prediction of driver intended path at intersections, IEEE Intelligent Vehicles Symposium Proceedings, 2014.

[10] Hinton, G. E.; Sejnowski, T. J. D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, eds. Learning and Relearning in Boltzmann Machines, 1986.

[11] Hinton, G. E.; Osindero, S.; Teh, Y. A fast learning algorithm for deep belief nets Neural Computation. 18 (7): pp.1527–1554, 2006.

[12] Ratul Mahajan, CRAWDAD dataset microsoft/vanlan (v. 2007-09-14), downloaded from http://crawdad.org/microsoft/vanlan/20070914, https://doi.org/10.15783/C7FG6S, Sep 2007.

[13] I. Tal, G. M. Muntean, User oriented fuzzy logic based clustering scheme for vehicular Ad hoc networks, IEEE Vehicular Technology Conf. (VTC Spring), pp. 1–5, 2013.

[14] Bevish Jinila, Komathy Rough Set Based Fuzzy Scheme for Clustering and Cluster Head Selection in VANET ELEKTRONIKA IR ELEKTROTECHNIKA, ISSN 1392-1215, Vol. 21, No. 1, 2015.

[15] Miki Aoyagi Learning Coefficient in Bayesian Estimation of Restricted Boltzmann Machine, Journal of Algebraic Statistics Vol. 4, No. 1, pp. 31-58, 2013.

APPENDIX

Code Segment:

```
 Communication Prototype Functions */
#include <mspnvehicle.h>
#include <killApp.h>
#include <boost/thread/thread.hpp>
void openNewTerminal();
bool notificationKilledProcess
(atv_acrosser::killApp::Request &req,
void receiveKillCommunication(int argc, char **argv);
boost::mutex mtxTerminal;
boost::mutex::scoped_lock lock (mtxTerminal);
boost::condition_variable condTerminal;
/* It proves the presence of
terminal window that execute the process
communication */
bool existTerminal = false;
/*****************************************************************
* @function: openNewTerminal
* Thread opens a new terminal and executes the communication
* program. It also remains waiting status on the condition variable
* to launch again the process communication.
*****************************************************************/
void openNewTerminal()
{
int statusSystem = 0;
/*Open the first terminal with communication program*/
existTerminal = true;
statusSystem = system("gnome-terminal -x ./communication");
printf("\nTERMINAL_OPENED_STATUS: %d", statusSystem);
/* Infinite while, there will be always a condition variable
which wait a signal from a killed process.
When the the condition variable will be awake from a killed process,
it will open a new terminal and execute the communication
program and wait again another signal from a killed process */
while(1)
{/
*Condition variable, wait to be awake after the killed process */
while(existTerminal == true)condTerminal.wait(lock);
/*Open a new terminal and execute the communication process */
statusSystem = system("gnome-terminal -x ./communication");
printf("\nTERMINAL_OPENED_STATUS: %d", statusSystem);
if(statusSystem < 0)
 printf("\n_PROBLEM_TO_OPEN_THE_NEW_WINDOW_DURING_THE
RESTARTING_OF_THE_SOFTWARE_communication");
```

```
}}
*****************************************************************
* @function: receiveKillCommunication
* Thread waits the communication with communication process via
* ROS service in case the process communication needs to
* terminate. When receive the notice from the service the
* function notificationKilledProcess is called.
*****************************************************************/
void receiveKillCommunication(int argc, char **argv)
{
ros::init(argc, argv, "");
ros::NodeHandle n;
//Here the service called
"restartCommunication" is created and
//advertised over ROS.
ros::ServiceServer service = n.advertiseService
("restartCommunication", notificationKilledProcess);
 ros::spin();
}/
**********
*************************
*****************************
* @function: notificationKilledProcess
* This function has called each time
that ROS service answers from
* the communication creating * a syncronization with it.
* The function will change in false
the value of the variable
* existTerminal and wake up the
* condition variable condTerminal
* with the scope of open a
new terminal and execute the process
* communication.
*****************************************************************/
bool notificationKilledProcess(atv_acrosser::killApp::Request &req,
atv_acrosser::killApp::Response &res)
{
 ROS_INFO("PID KILLED %ld", (long int)req.pid2Kill);
/* set to false the variable existTerminal, it means there are nt
open terminal with running communication */
existTerminal = false;
/* wake up the condition variable condTerminal */
 condTerminal.notify_one();
return true;
}
int main(int argc, char **argv)
```

```
{
boost::thread openNewTerminal_Thread(&openNewTerminal);
boost::thread receiveKillCommunication_Thread(
&receiveKillCommunication, argc, argv);
openNewTerminal_Thread.join();
receiveKillCommunication_Thread.join();
return 0;
 }
/* Boltzmann Prototype with Fuzzy training*/
#include <math.h>
#include <fstream>
#include <iostream>
#include <random> using namespace arma;using namespace
std;
#define elif else if
#define HIDDEN_SIZE 200
#define BATCH_SIZE 2000oncatenateMat(vector<mat> &vec){int
height = vec[0].n_rows;int width = vec[0].n_cols;
mat res = zeros<mat>(height * width, vec.size());
for(int i=0;
i<vec.size(); i++){mat img = vec[i];
img.reshape(height * width, 1);res.col(i) = img.col(0);}
res = res / 255.0;return
res;}int ReverseInt (int i){
unsigned char ch1, ch2, ch3, c
h4;ch1 = i & 255;c
h2 = (i >> 8) & 255;ch3 = (i >> 16) &
255;ch4 = (i >> 24) & 255;
return((int) ch1 << 24) +
((int)ch2 << 16) + ((int)ch3 << 8) + ch4;}
void read_Mnist(string
filename, vector<mat> &vec)
{ifstream file (filename, ios::binary);
if (file.is_open()){int magic_number = 0;int
number_of_images = 0;
int n_rows = 0;int n_cols = 0;
file.read((char*) &magic_number,
sizeof(magic_number));
magic_number = ReverseInt(magic_number);
file.read((char*)
&number_of_images,sizeof(number_of_images));
number_of_images = ReverseInt(number_of_images);
file.read((char*)
&n_rows, sizeof(n_rows));
n_rows = ReverseInt(n_rows);
file.read((char*) &n_cols, sizeof(n_cols));n_cols =
```

```
ReverseInt(n_cols);
for(int i = 0;
i < number_of_images; ++i){mat tp(n_rows, n_cols);
for(int r = 0; r < n_rows; ++r)
{for(int c = 0; c < n_cols; ++c)
{unsigned char temp = 0;file.read((char*)
&temp, sizeof(temp));tp(r, c) = (double)
temp;}}vec.push_back(tp);}}}
voidreadData(mat &x, string xpath)
{//read MNIST iamge into Arma Mat
vectorvector<mat> vec;read_Mnist(xpath, vec);
random_shuffle(vec.begin(), vec.end());
x = concatenateMat(vec);}mat
sigmoid(mat M){return 1.0 / (exp(-M) + 1.0);}
voidmatRandomInit(mat &m, int rows,
int cols, double scaler){m =
randn<mat>(rows, cols);m = m * scaler;
}matgetBernoulliMatrix(mat &prob)
{// randu builds a Uniformly distributed
matrixmat ran = randu<mat>
(prob.n_rows, prob.n_cols)
;mat res = zeros<mat>(prob.n_rows,
prob.n_cols);res.elem(find(prob > ran)).
ones();return res;}
voidsave2txt(mat &data, string str, int step){string s =
std::to_string(step);str += s;
str += ".txt";FILE *pOut = fopen(str.c_str(), "w");
for(int i=0; i<data.n_rows; i++){for(int
j=0; j<data.n_cols; j++)
{fprintf(pOut, "%lf", data(i, j));
if(j == data.n_cols - 1) fprintf(pOut, "\n");
else fprintf(pOut, "
");}}fclose(pOut);}
matFCBM_training(mat x, int hidSize,
int batchSize, int cd_k) /* Fuzzy Numbers*/
{int nfeatures = x.n_rows;
int nsamples = x.n_cols;
// b is hidden layer;// c is visible layermat
w, b, c;matRandomInit(w,
nfeatures, hidSize, 0.12);matRandomInit(b, hidSize, 1, 0);
matRandomInit(c, nfeatures, 1, 0);int counter = 0;double
lrateW = 0.01; //Learning rate fo
r weights double lrateC = 0.01; /
/Learning rate for biases of visible units double lrateB
= 0.01; //Learning rate for biases
of hidden units
```

```
double weightcost = 0.0002; double initialmomentum = 0.5;double
finalmomentum = 0.9;double
errsum = 0.0;double momentum;
mat incW = zeros(w.n_rows, w.n_cols);mat incB =
zeros(b.n_rows, b.n_cols);
mat incC = zeros(c.n_rows, c.n_cols);
while(1){// start positive phaseint randomNum =
((long)rand() + (long)rand()) %
(nsamples - batchSize)
;mat data = x.cols(randomNum, randomNum + batchSize -
1);data = getBernoulliMatrix(data);
mat poshidprobs = sigmoid(w.t() * data + repmat(b, 1,
batchSize));poshidprobs =
normalise(poshidprobs, 1, 0);
mat posprods = data * poshidprobs.t() /
batchSize;mat poshidact = sum(poshidprobs, 1) /
batchSize;mat posvisact = sum(data, 1)
/ batchSize;// end of positive
phasemat poshidprobs_temp = poshidprobs;mat
poshidstates, negdata;
// start negative phase//
CD-K algfor(int i = 0; i < cd_k; i++){poshidstates =
getBernoulliMatrix(poshidprobs_temp);
negdata = sigmoid(w * poshidstates + repmat(c, 1, batchSize));
negdata =getBernoulliMatrix(negdata);
poshidprobs_temp = sigmoid(w.t() * negdata + repmat(b, 1, batchSize));
poshidprobs_temp = normalise(poshidprobs_temp, 1, 0);}
mat neghidprobs = poshidprobs_temp;
mat negprods = negdata * neghidprobs.t() /
batchSize;mat neghidact = sum(neghidprobs, 1)
/ batchSize;mat negvisact = sum(negdata, 1) / batchSize;//end of
negative phasedouble err = accu(pow(mean(data - negdata, 1), 2.0));
//errsum = err + errsum;if(counter > 10) momentum
= finalmomentum;
else momentum = initialmomentum;
// update weights and biasesincW = momentum * incW + lrateW
* ((posprods - negprods) - weightcost * w);
incC = momentum * incC + lrateC * (posvisact - negvisact);
incB =
momentum * incB + lrateB * (poshidact - neghidact);
w += incW;c += incC;b += incB;cout<<"counter = "
"<<counter<<", error = "<<err<<endl;
if(counter % 100 == 0){save2txt(w, "w/w_", counter / 100);
save2txt(b, "b/b_",
counter / 100);save2txt(c, "c/c_", counter / 100);}
if(counter >= 10000) break;++ counter;}
```

```
return w;}int main(int argc,
char** argv){long start, end;start = clock();
mat trainX;readData
(trainX, "mnist/train-images-idx3-ubyte");cout<<"Read
trainX successfully,
including "<<trainX.n_rows<<"
features and "<<trainX.n_cols<<" samples."<<endl;// Finished
reading datamat
w = FCBM_training(trainX, HIDDEN_SIZE, BATCH_SIZE, 1);
end = clock();cout<<"Totally used
time: "<<((double)(end - start)) /
CLOCKS_PER_SEC<<" second"<<endl;return 0;
```